Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (currently amended): An apparatus for transferring data in a telecommunications network comprising:

a memory in which a packet memory length is stored;

means for determining a link length for the packet that will be sent into the network based on the memory length and at least one parameter, where the link length correctly corresponds to the packet's physical line bandwidth usage in the network, the determining means includes a controller, the controller includes a billing module for billing for the physical line bandwidth usage by the packet based on the link length of the packet; and

means for sending the packet having the link length to the network.

Claim 2 (original): An apparatus as described in Claim 1 including a scheduler for scheduling when the packet having the link length is to be sent into the network based on the link length of the packet.

Claims 3 and 4 (canceled)

Claim 5 (currently amended): An apparatus as described in Claim [[4]] <u>2</u> wherein the determining means determines the link length according to

$$LinkLength = \left\lceil \frac{\{MemoryLength - HeaderSize + MPLSAdj\}}{FragmentSize} \right\rceil \times \{HeaderSize + FrameOverhead\} + \{MemoryLength - HeaderSize + MPLSAdj\} + LastFragmentPad$$

Where,

LinkLength	=	Length of the packet as seen on the physical line
MemoryLength	=	Length of the packet as stored in the internal memory (= Header + Payload)
HeaderSize	=	Length of the Header for this packet
MPLSAdj	=	MPLS label push (positive value) / pop (negative value) amount
FragmentSize	=	Required payload size of each packet fragment
FrameOverhead	=	Number of bytes that will be added to each frame/packet in SONET/SDH framing
LastFragmentPad	=	Number of bytes that need be added to the last fragment of a fragmented packet to make the fragment minimum size

Claim 6 (original): An apparatus as described in Claim 5 wherein the link length is determined by the controller if the packet is an ATM cell according to

$$LinkLength = MemoryLength + FrameOverhead.$$

Claim 7 (original): An apparatus as described in Claim 6 including a port card having the memory, determining means, sending means, scheduler and billing module.

Claim 8 (original): An apparatus as described in Claim 7 wherein the memory includes a packet storage handler in communication with the billing module and the scheduler.

Claim 9 (currently amended): An apparatus as described in Claim 8 wherein the sending means includes a packet framing[[.]] and line interface in communication with the packet storage handler.

Claim 10 (original): An apparatus as described in Claim 9 wherein the sending means includes a fragmentation/segmentation/encapsulation/decapsulation module in communication with the packet storage handler and the packet framing and line interface.

Claim 11 (original): An apparatus as described in Claim 10 wherein the controller frees the memory based on the memory length of the packet that has been sent into the network.

Claim 12 (original): An apparatus as described in Claim 11 wherein the controller determines memory length according to

$$MemoryLength = LinkLength + HeaderSize \mp MPLSAdj - LastFragmentpad$$
$$- [NumFragments \times (HeaderSize + FrameOverhead)]$$

Where

$$NumFragments = \left\lceil \frac{LinkLength}{(HeaderSize + FragmentSize + FrameOverhead)} \right\rceil.$$

Claim 13 (currently amended): A method for transferring data in a telecommunications network comprising the steps of:

storing in a memory a memory length of a packet;

determining a link length for the packet that will be sent into the network based on the memory length and at least one parameter, where the link length correctly corresponds to the packet's physical line bandwidth usage in the network; [[and]]

sending the packet having the link length to the network[[.]]; and

billing for the physical line bandwidth usage by the packet based on the link length of the packet.

Claim 14 (original): A method as described in Claim 13 including the step of scheduling when the packet having the link length is to be sent into the network based on the link length of the packet.

Claim 15 (canceled)

Claim 16 (currently amended): A method as described in Claim [[15]] 14 wherein the determining step includes the step of determining the link length according to

$$LinkLength = \left\lceil \frac{\{MemoryLength - HeaderSize \pm MPLSAdj\}}{FragmentSize} \right\rceil \times \{HeaderSize + FrameOverhead\} + \{MemoryLength - HeaderSize \pm MPLSAdj\} + LastFragmentPad$$

Where,

LinkLength	=	Length of the packet as seen on the physical line
MemoryLength	=	Length of the packet as stored in the internal memory (= Header + Payload)
HeaderSize	=	Length of the Header for this packet
MPLSAdj	=	MPLS label push (positive value) / pop (negative value) amount
FragmentSize	=	Required payload size of each packet fragment
FrameOverhead	=	Number of bytes that will be added to each frame/packet in SONET/SDH framing
LastFragmentPad	=	Number of bytes that need be added to the last fragment of a fragmented packet to make the fragment minimum size.

Claim 17 (original): A method as described in Claim 16 wherein the determining step includes the step of determining the link length for an ATM cell according to

$$LinkLength = MemoryLength + FrameOverhead.$$

Claim 18 (original): A method as described in Claim 17 wherein the sending step includes the step of sending the packet to the network through a port card having the memory.

Claim 19 (original): A method as described in Claim 18 including the step of freeing the memory based on the memory length of the packet that has been sent into the network.

Claim 20 (original): A method as described in Claim 19 wherein the freeing step includes the step of determining memory length according to

$$MemoryLength = LinkLength + HeaderSize \mp MPLSAdj - LastFragmentpad \\ -[NumFragments \times (HeaderSize + FrameOverhead)]$$

Where,

$$NumFragments = \left\lceil \frac{LinkLength}{(HeaderSize + FragmentSize + FrameOverhead)} \right\rceil.$$